BRL

AD 78073

CW-

REPORT NO. 1718

DYNAMIC STORAGE ALLOCATION FOR
THE BRLESC II COMPUTER

Morton A. Hirschberg

May 1974

USA BALLISTIC RESEARCH LABORATORIES
ABERDEEN PROVING GROUND, MARYLAND

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BRL REPORT NO. 1718	**2. GOVT ACCESSION NO.**	**3. RECIPIENT'S CATALOG NUMBER**
4. TITLE *(and Subtitle)* DYNAMIC STORAGE ALLOCATION FOR THE BRLESC II COMPUTER		**5. TYPE OF REPORT & PERIOD COVERED** FINAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Morton A. Hirschberg		**8. CONTRACT OR GRANT NUMBER(s)**
9. PERFORMING ORGANIZATION NAME AND ADDRESS USA Ballistic Research Laboratories Aberdeen Proving Ground, MD 21005		**10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS**
11. CONTROLLING OFFICE NAME AND ADDRESS US Army Materiel Command 5001 Eisenhower Avenue Alexandria, VA 22304		**12. REPORT DATE** MAY 1974
		13. NUMBER OF PAGES 32
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		**15. SECURITY CLASS. *(of this report)*** UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Dynamic Storage Allocation	List Processing	List Manipulation
Dynamic Storage	Free Storage	Random Access
Linked List	Automatic Random Access	
Datasets	Data Manipulation	

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The use of dynamic storage allocation for the BRLESC II computer is described, as well as the use of linked lists. This system was fashioned after the dynamic storage scheme used in SIMSCRIPT. Some of the SIMSCRIPT names being quite descriptive have been used here.

DD FORM 1 JAN 73 **1473** EDITION OF 1 NOV 6S IS OBSOLETE

TABLE OF CONTENTS

I. INTRODUCTION

This report describes a dynamic storage allocation system for the
BRLESC II computer. Dynamic storage allocation is the automatic assign-
ment and release of storage during execution. It differs from conven-
tional programming by eliminating the necessity of requiring storage
of many seldom-used arrays. Dynamic storage allows the programmer to
make more efficient use of available storage, even to the extent of
reducing the requirements for chain or overlay jobs.

The system comprised of over forty FORTRAN subroutines called by a
user programmer is fashioned after the dynamic storage scheme used in
SIMSCRIPT (a simulation language). Some of the SIMSCRIPT function
names being quite descriptive have been carried over into this system.

The major system features provide for automatic data overflow onto
peripheral storage, linked list capability, and garbage collection.

Automatic overflow is from high speed memory to the disc, which
appears to the user as an extension of high speed memory. No special
programming is required to make use of it, although the programmer has
the option to write data onto the disc if he so desires.

A linked list is one such that some of its elements are themselves
lists. Lists are composed of datasets (storage cells containing
related information), which are the basic information unit of the system.
A dataset may itself become a list or several linked together to form a
list. Datasets in use are located in core storage, those not in use,
are transferable to the disc. The system automatically transfers unused
datasets to the disc when their space is required by other datasets.
The system does allow the user to lock a dataset in core storage so that
it may not be transferred to the disc; however, this feature should be
used with descretion or an early termination could occur. There is no
limit as to the number of datasets which may be created, or to the
number of lists to which a dataset may belong. The system does not
however determine to which lists a dataset belongs. Because of this, it
is the users responsibility to insure that a dataset belongs to no lists
when a dataset release request is made.

Datasets may be any length and need not belong to a list (if they
provide temporary storage for intermediate calculations). The system
allows a dataset to be placed anywhere within a list or placed in order
according to a specified sort word in the dataset. List searching
routines exist to locate particular datasets in the list. Lists may be
searched from top to bottom or bottom to top. More rapid access to
datasets appearing on lists in also available and will be described in
Section III. C. 16.

Garbage collection is the process by which storage may be reclaimed
for a newly created dataset by sweeping through memory identifying only

5

those structures currently referenced by the program. This occurs only after all free storage has been depleted. Garbage collection is time consuming but frees the programmer from performing detailed bookkeeping which would otherwise be necessary.

The Dynamic Storage Allocation System resides on the BRL DISC and may be accessed through the *COMPILE card as follows:

*COMPILE DISC, DAS, ALL

Dynamic storage systems have been written for many computers, however, this particular system is easily tranferable to computers whose computer word length is at least 60 bits (i.e. CDC computers).

Dynamic storage allocation has found widespread use in time and event based simulation programs although the latter usage is more frequent. Proposed uses of dynamic storage include large problems with fixed databases having infrequently used subsets. Serial or linear data processing may render greater efficiency to dynamic storage, although careful thought is required in any large database program. Dynamic storage is applicable for any queuing problem.

II. SYSTEM DESCRIPTION

A. Storage Allocation

The user may designate available storage to occupy from one to ten storage units. The first of these is assumed to be core storage. The user must specify the length available to the system. Excluding user code, the system including the disc routines require 10,000 words. Currently, the system uses only one other unit besides core storage, the disc. Approximately 600,000 words are available to the user without special handling.

The system sets up storage requested on each unit as a free block and links the blocks together to form a free list. When a block is requested, the free list is searched until a free block of adequate size if found. The necessary storage is then allocated from the lower portion of this block, and the remainder made into a smaller free block. When storage is returned to the system, it is added to an adjacent free block, or if this is not possible, it is added to the free list as a separate block.

When a request is made for a block larger than any free block, a garbage collection is initiated in an attempt to relocate datasets within memory so that all free blocks are brought together to form one free block. If this procedure does not produce enough storage for the requested block, a search is made for a block which can be moved to the disc. Movable datasets are written onto the disc and their space made available to the free list. If space still cannot be found, a message is written and execution terminated. Garbage collection only occurs for core storage.

6

B. Lock Numbers for Datasets

The system determines whether a dataset is movable or not by examining the lock number of the dataset. Lock numbers range from 0 to 7. Lock numbers from 0 through 5 are assigned by the user to datasets he creates.

NUMBER	MEANING
0	Unlocked. May be moved to provide room for a newly requested dataset.
1	Locked and currently in use. Will not be moved until it is no longer in use.
2-4	Locked. Will not be moved for any reason. User may set these levels for his own purposes. The system does not distinguish between levels 2-4.
5	Locked and indestructible. It will not be moved or released to the system.
6	Free.
7	Special system designated datasets for storage of one-word system parameters.

C. Lists

A list is a set of logically associated datasets connected by means of system-created link words. Lists are used as a means to quickly locate datasets sharing common attributes. A dataset may belong to any number of lists.

1. List Header Variable

A list header variable is one word stored within a dataset, locally, or in user common. This variable allows the user to refer to the list. It must be defined. The contents of this variable point to the first and last link words in the list.

2. Link Words

The link word contains the address of the next link word of the list (or zero if the link is the last dataset of the list), the address of the previous link word of the list (or zero if the link is the first dataset of the list), and the address of the dataset pointer (DSP word). One such word exists for each dataset in the list.

3. Dataset Pointer (DSP word).

The DSP word contains the current address of the dataset and the number of words in the dataset. One such word exists for each dataset.

III. SYSTEM UTILIZATION

A. Initialization

1. Common Variables. The user must define certain necessary system variables in blank common and assign values to these variables. To see the necessary dimension, equivalence, and integer statements as well as the common definition, refer to the sample program given in Section III.H. The following system variables must be assigned initial values by the user's program.

QNAREA	The numbers of storage units to be used by the system. The value to use here is 2 indicating core storage and the disc.
QWAREA	The size in words of the ith storage unit (i.e. core storage = 5000, disc storage = 50,000). QWAREA is a one dimensional integer array of ten elements. At BRL, only words 1 and 2 of this array need be set. Word 1 is for core storage, and word 2 for the disc.
QDSIZE	A one dimensional integer array specifying the incremental size of datasets. This array consists of 10 elements. At least one specification must be made (it may be zero).

For example: QDSIZE(1) = 5
                QDSIZE(2) = 10
                QDSIZE(3) = 0

The above statement indicates that no dataset will be smaller than 5 words; datasets with requested sizes from 1-4 words will automatically be allocated 5 words (one location is required by the system as a zeroth word). Dataset requests of lengths 5-9 will be allocated 10 words, etc. Values must be assigned in ascending order. When a value of zero is assigned, this implies that the exact number of words required will be allocated. In the above example any dataset request for 10 or more words will be allocated the exact number of words necessary.

QERLUN   The logical variable containing the unit number
of the device to be used to print system error
messages. At BRL, QERLUN should be set to 6.

2. <u>Subroutine QINITL</u>   After blank common has been set, the
user must call QINITL to initialize other system variables and to set
up the free list.

B. <u>Dataset Creation and Release</u>

Three subroutines exist for the creation of datasets, and one for
the release of datasets. The first three routines discussed below
clear the contents of the dataset created to zeros.

1. <u>Subroutine CREATE</u>

CREATE (N, INDEX) where
N = integer expression
INDEX = integer variable name

This subroutine creates a dataset of a least N+1 words (space
allocated depends upon the values assigned to the QDSIZE array). The
dataset INDEX (pointer to the first word of the dataset) is returned
as the value of the second argument. CREATE sets the lock value of
the dataset to zero (unlocked) and creates a DSP word, indicating that
the dataset will be part of a list.

2. <u>Subroutine CREATL</u>

CREATL (N, INDEX)

CREATL performs the same functions as the subroutine CREATE except
that the lock number is set to 1 (locked). The definitions of N and
INDEX are the same as for subroutine CREATE.

3. <u>Subroutine CREATX</u>

CREATX (N, INDEX)

CREATX creates a dataset of length N+1 (regardless of the limits
specified by the QDSIZE array) and sets the lock value to 1. No DSP
word is created, indicating that the dataset is not part of any list;
i.e. the dataset is intended for use as a temporary storage area. The
definitions of N and INDEX are the same as for subroutine CREATE.

9

4. Subroutine DSTROY

DSTROY (INDEX)

This subroutine returns the space allocated to the dataset whose INDEX is the argument specified. This subroutine does not modify any list linkages to which this dataset may be a part. The user must do this with a call of REMOVE (see Section III.C.13.).

C. List Manipulation

These subroutines allow the user to create, modify, or destroy lists.

1. Subroutine PUTTOP

PUTTOP (LIST, INDEX) where
LIST = variable name
INDEX = integer variable name.

This subroutine places the dataset INDEX (supplied by CREATE, CREATL or CREATX) as the first dataset of the list indicated by the list header variable LIST. The list header variable may be thought of as a list name; it contains the addresses of the first and last link words in the list. If the list already contains datasets, the dataset will be inserted as the first dataset and list linkages will be changed wherever necessary to reflect this modification.

2. Subroutine PUTBOT

PUTBOT (LIST, INDEX)

PUTBOT functions in the same manner as PUTTOP except that the named dataset is inserted as the last dataset in the list.

3. Subroutine PUTORA

PUTORA (LIST, INDEX, N) where
LIST = variable name
INDEX = integer variable name
N = integer expression

PUTORA inserts dataset INDEX into list LIST according to the value of the work N within the dataset. The dataset will be inserted such that it follows the dataset, if any, which has a lesser value in its Nth word position, and precedes the dataset, if any, which has a greater value in its Nth word position. In other words, PUTORA creates a list sorted in ascending order according to some word within the datasets. The value of the word must be a real value.

10

4. Subroutine PUTOIA

    PUTOIA (LIST, INDEX, N)

PUTOIA functions in the same manner as PUTORA except that the value
in the Nth word position is an integer value.

5. Subroutine PUTORD

    PUTORD (LIST, INDEX, N)

PUTORD functions in the same manner as PUTORA except that the
dataset INDEX is inserted such that the values in the Nth word position
are in descending order. The value in the Nth word position is a real
value.

6. Subroutine PUTOID

    PUTOID (LIST, INDEX, N)

PUTOID functions in the same manner as PUTORD except that the
value in the Nth word position is an integer value.

7. Subroutine PUTAFT

    PUTAFT (LIST, LINK, INDEX 1, INDEX 2) where
    LIST = variable name
    LINK = variable name
    INDEX 1 = integer variable name
    INDEX 2 = integer variable name

PUTAFT allows the user to insert dataset INDEX 2 into the list
LIST after the dataset specified by INDEX 1. The link parameter LINK
contains the contents of the link word for the dataset specified by
INDEX 1.

8. Subroutine PUTBEF

    PUTBEF (LIST, LINK, INDEX 1, INDEX 2)

PUTBEF functions similarly to PUTAFT except that the dataset
specified by INDEX 2 is placed before the dataset specified by INDEX 1.

9. Subroutine NEXT

    NEXT (LINK, INDEX) where
    LINK = variable name
    INDEX = integer variable name

11

NEXT allows the user to search forward through a list. The link parameter LINK must initially be set to the value of the list header variable. After NEXT has been executed, the link variable contains the contents of the current link word. The variable INDEX is returned as the index of the next dataset in the list or zero if no further datasets are on the list. Typical use of NEXT is as follows:

```
      KLINK = KFILE
10    CALL NEXT (KLINK, INDX)
      IF (INDX. EQ. 0) GO TO 20

      (OPERATIONS ON DATASET INDX)

      GO TO 10


20 CONTINUE
```

NEXT also sets and resets lock numbers of datasets. If the lock number of the current dataset is 0, it is reset to 1 to indicate that the dataset is currently in use. The lock value of the previous dataset is set to 0 if it was 1, indicating that the dataset is no longer in use. Datasets with lock numbers greater than 1 are not set and reset but remain locked throughout the execution of the subroutine.

The set-reset feature is provided so that there is no possibility that a dataset will be moved if the loop performing the search contains another list-searching loop, or contains routines calling for the creation of new datasets.

10. Subroutine NEXTNL

    NEXTNL (LINK, INDEX)

NEXTNL functions in the same manner as NEXT except that the lock values are not set and reset.

11. Subroutine PREV

    PREV (LINK, INDEX)

PREV functions in the same manner as NEXT except that the list is searched backward beginning with the last dataset in the list.

12. Subroutine PREVNL

    PREVNL (LINK, INDEX)

PREVNL functions in the same manner as PREV except that lock values are not set and reset.

13. **Subroutine REMOVE**

    REMOVE (LIST, LINK, INDEX)

REMOVE removes the dataset INDEX from list LIST. The link parameter LINK is the link word as returned by NEXT, NEXTNL, PREV, OR PREVNL. After removing the dataset, REMOVE replaces the value of the link parameter with the previous link word so that a NEXT or NEXTNL loop may be continued. If the link value is not available to the user at the time he wishes to remove a dataset from a list, the value of 0 may be used as a link parameter. In this instance, the link parameter will not be updated.

14. **Subroutine REMOVP**

    REMOVP (LIST, LINK, INDEX)

REMOVP functions in the same manner as REMOVE except that, for LINK≠0, the next link toward the bottom of the list is returned as the value of the link parameter (for continuation of a PREV or PREVNL loop).

15. **Subroutine WIPOUT**

    WIPOUT (LIST, KDSTRY) where
    LIST = variable name
    KDSTRY = integer expression

WIPOUT removes all datasets from the list LIST (list header variable). If the value of KDSTRY is one, the datasets will be destroyed as well as removed from the list.

16. **Subroutines DSPWRD and INDWRD**

These routines provide access to a dataset more rapidly than using a sequential search. It involves storing DSP word addresses of those datasets for which quick access is desired.

    DSPWRD (INDEX, IDSPWD) where
    INDEX = integer variable name
    IDSPWD = variable name

DSPWRD returns the location of the DSP word for dataset INDEX in the variable specified by IDSPWD. A DSP word for a dataset resides in the same memory location throughout the existence of a dataset. By storing the location of the DSP word for the dataset at the time it is created, the current index of the dataset may be retrieved as follows:

INDWRD (IDSPWD,INDEX)

INDWRD returns the current index INDEX of the dataset whose DPS word address is stored in IDSPWD. If the dataset is currently located on the disc, it is loaded into core storage and the index then returned.

D. Lock Number Manipulation

1. Subroutine LOCKDS

LOCKDS (INDEX, LOCKNR) where
INDEX = integer variable name
LOCKNR = integer expression

LOCKDS sets the lock number of dataset INDEX to the value specified by LOCKNR, an integer value ≤ 5.

2. Subroutine NLOKDS

NLOKDS (INDEX)
NLOKDS unlocks dataset INDEX.

3. Subroutine LOCKFL

LOCKFL (LIST, LOCKNR)

LOCKFL sets the lock number for each dataset in list LIST (list header variable) to the value specified by LOCKNR (≤ 5).

4. Subroutine NLOKFL

NLOKFL (LIST, LOCKNR)

NLOKFL unlocks all datasets in the list LIST (list header variable) which have a lock number less than or equal to the value of LOCKNR (an integer ≤ 5).

E. Accessing Data Within Datasets

The last variable in blank common is the one-dimensional, one element array Q. This word is the first word of the core storage area requested by the user; the rest of the area follows in contiguous locations. Because the compiler does not check for subscript references larger than those defined, Q allows the user to refer to any word within a dataset. The dataset index refers to the position of the first word of the dataset within the array Q. To refer to the first word of data within a dataset specify:

Q (INDEX)

To refer to the Nth word, specify

Q (INDEX - 1+N).

To allow reference to real and integer values as well as using two subscripts rather than one, the IQ(1), QQ(1,1), and IQQ(1,1) variables are all made equivalent to the variable Q(1). To use the two dimensional forms specify the index as the first subscript and the position of the particular word within the dataset as the second subscript. For example, to refer to the 5th word of the dataset whose index is INDX specify:

QQ (INDX,5).

In the interest of program readability another method of referring to dataset values is available. This is accomplished by the user equivalencing variables to the Q array. The equivalence block must appear in all routines which refer to the dataset. It will be noted that one dimension is added to each variable to account for the dataset index. This limits those variables which can be referred to through equivalent forms to two-dimensional variables.

F.   Dataset Manipulations

1.   Subroutines DSLNTH

DSLNTH (INDEX, LENTH)

DSLNTH returns the length LENTH of dataset INDEX.   LENTH does not include the zeroth word.

2.   Subroutine DSXPND

DSXPND (INDEX, NULNTH)

DSXPND expands dataset INDEX to accommodate NULNTH + 1 words. If free storage exists below the dataset, it is given to the dataset. If not, a new block is created and the data transmitted to it.   The old dataset is destroyed, but its DSP word is kept for the new dataset.

3.   Subroutine PUTDRM

PUTDRM (INDEX)

PUTDRM writes dataset INDEX onto the disc.

G.   System Restrictions

Blank common other than that defined for system use may not be used by the user's program.

The following variable names are reserved and may not otherwise be used by those routines which need system blank common:

QNAREA	QDSIZE
QWAREA	QNSIZE
QFREHD	QLUNIT
QNLNKS	QFBITS
QZSIZE	Q
QNZBLK	IQ
QZHEAD	QQ
QCOUNT	IQQ

The follwoing are system subroutine reserved names:

BITX	PREV	QFLDST
CREATE	PREVNL	QGCBLK
CREATL	PUTAFT	QGDBLK
CREATX	PUTBEF	QGRBAG
DSADMP	PUTBOT	QGTZWD
DSLNTH	PUTDRM	QINITL
DSPWRD	PUTOIA	QPTZWD
DSTROY	PUTOID	QZBLOK
DSXPND	PUTORA	RDRUM
INDWRD	PUTORD	REMOVE
LISPRT	PUTTOP	REMOVP
LOCKDS	QCEASE	RITEA
LOCKFL	QCREAT	RITEF
NEXT	QDSRED	RITEI
NEXTNL	QDSRYT	RITEO
NLOKDS	QDSTRY	UNPX
NLOKFL	QERROR	WDRUM
PACX	QFIELD	WIPOUT
		XMIT

H. Sample Program

This section contains a payroll update program example using all of the system features, though not all of its subroutines.

Problem:

1. Read in the following employee data.

   a. Employee number
   b. Employee name
   c. Salary
   d. Job assignment data

2. For some of the above employees the following update information.

   a. Employee number
   b. New salary

3. Prepare a list of employees, highest salary first with employee number, name and salary.

The following pages contain a listing of the program cards, the input data, and the output.

```
PROGRAM EXAMPLE

COMMON          QNAREA, QWAREA(10), QFREHD, QNDTST, QNLNKS,
1       QZSIZE, QNZBLK, QZHEAD, QCOUNT(30), QDSIZE(10),
2       QNSIZE, QLUNIT(10), QERLUN, QFBITS(2, 10),
3       Q(1)
   DIMENSION IQ(1), QQ(1,1), IQQ(1, 1)
   EQUIVALENCE (Q(1),IQ,QQ,IQQ), (QCRSIZ, QWAREA(1))
   INTEGER QNAREA, QWAREA, QFREHD, QNDTST, QNLNKS, QZSIZE,
1    QNZBLK, QCOUNT, QDSIZE, QMODLK, QNSIZE, QLUNIT,
2    QFIELD, QERLUN, QFBITS, QCRSIZ
C
C

   DIMENSION NTEMP (2) TEMP (12)
C
C   --------------- THE FOLLOWING IS AN EQUIVALENCE BLOCK FOR USE IN
C   --------------- REFERRING TO THE DATA IN THE EMPLOYEE INFORMATION
C   --------------- DATASET.

   EQUIVALENCE (Q(1), EMPLNR), (Q(2),NAME),  (Q(4), SALARY),
1    (Q(5), WRKDAT)
   DIMENSION EMPLNR(1), NAME(1,2), SALARY(1), WRKDAT(1,3,4)
   INTEGER EMPLNR, WRKDAT
C
C
   --------------- INITIALIZE SYSTEM.  HIGH SPEED MEMORY STORAGE SIZE
C   --------------- IS 5000 WORDS, AND DISC SIZE IS 50000.  NO
C   --------------- MINIMUM SIZE OF DATASETS.

   QNAREA = 2
   QWAREA(1) = 5000
   QWAREA(2) = 50000
   QDSIZE(1) = 0
   QLUNIT(1) = 20
   QERLUN = 6
   CALL QINITL
C
C   --------------- READ IN MAXIMUM EMPLOYEE NUMBER.   CREATE A
C   --------------- LOCKED DATASET OF THAT LENGTH FOR STORAGE OF DSP
C   --------------- WORD ADDRESSES
```

17

```
      READ 1, NREMPL
    1 FORMAT(I10)
      CALL CREATL (NREMPL, INDDSP)
C
C     -------------- READ EMPLOYEE DATA.  (ONE EMPLOYEE AT A TIME)
C
   10 READ 2, JFLAG, NEMPL, (NTEMP(I),I=1,2), SALRY, (TEMP(I),I=1,12)
    2 FORMAT (A6, I4, 2X, 2A10, 2X, F10.2, 2X, 12I2)
      IF (JFLAG .EQ. 6HENDDAT) GO TO 20
C
C     -------------- CREATE DATASET OF 16 WORDS AND STORE DATA
C
      CALL CREATE (16, INDEX)
      EMPLNR (INDEX) = NEMPL
      CALL XMIT (2, NTEMP, NAME(INDEX,1))

      SALARY(INDEX) = SALARY
      CALL XMIT(12, TEMP, WRKDAT(INDEX, 1,1))
C
C     -------------- OBTAIN DSP WORD OF DATASET AND STORE IN INDDSP
C     -------------- DATASET

      CALL DSPWRD(INDEX, IQQ(INDDSP,NEMPL))
      GO TO 10
C
C     --------------= READ IN SALARY UPDATE INFORMATION
C
   20 READ 3, JFLAG, NEMPL, SALRY
    3 FORMAT(A6, I4, 2X, F10.2)
      IF(JFLAG .EQ. 6HENDDAT) GO TO 30
C
C     -------------- FIND DATASET ADDRESS USING DSP WORD
      CALL INDWRD(IQQ(INDDSP,NEMPL), INDEX)
      SALARY(INDEX) = SALRY
      GO TO 20
C
C     -------------- PREPARE LIST OF DATASETS ORDERED HIGHEST SALARY
C     -------------- FIRST FOR OUTPUT PURPOSES.  ZERO OUT LIST HEADER.
   30 LSTSAL = 0
      DO 40 J = 1,NREMPL
      IDSP = IQQ(INDDSP, J)
      IF(IDSP .EQ. 0) GO TO 40
      CALL INDWRD (IDSP, INDEX)
C
C     -------------- ORDER ON 4TH WORD, REAL DATA, IN DESCENDING ORDER
      CALL PUTORD (LSTSAL, INDEX, 4)
   40 CONTINUE
```

```
C
C        --------------- PRINT OUT LIST, RELEASE ALL STORAGE USED (THIS
C        --------------- IS NOT NECESSARY IN THIS EXAMPLE BUT IS DONE FOR
C        --------------- ILLUSTRATION)
C
      PRINT 4
    4 FORMAT(*1 EMPL. NR. NAME SALARY*//)
      LINK = LSTSAL
   50 CALL NEXTNL (LINK, INDEX)
      IF(INDEX .EQ. 0) GO TO 60
      PRINT 5, EMPLNR(INDEX), (NAME(INDEX,J), J=1,2), SALARY(INDEX)
    5 FORMAT(I15, 2X, 2A10, F10.2)
      CALL REMOVE (LSTSAL, LINK, INDEX)
      CALL DSTROY(INDEX)
      GO TO 50
C
C        --------------- RELEASE STORAGE USED BY INDDSP DATASET.
C
   60 CALL DSTROY(INDDSP)
C
      CALL EXIT
      END
```

The following is a list of the input cards to the sample program.

```
          75     NUMBER OF TOTAL COMPANY EMPLOYEES

          17 AARON, A.A              410.05
          28 BEACH, REDONDO          415.00
          48 CREEK, S. ANTONIO       623.20
           8 HEART, B.               438.35
          39 KILOWATT, R.            766.05
           3 MAYER, M.G.             912.15
          24 PINKHAM, L             1170.32
          16 RASPUTIN, I             806.50
          58 SMITH                   608.50
          12 TRUEHEART, T.          1216.33
          65 VAN PELT, LUCY          701.16
          47 ZWICK, Z.               607.11
ENDDAT
          12      1350.05
          16       815.50
           3       835.10
           8       450.25
          28       416.10

ENDDAT
```

OUTPUT FROM SAMPLE PROGRAM

EMPL. NR.	NAME	SALARY
12	TRUEHEART, T.	1350.05
24	PINKHAM, L.	1170.32
3	MAYER, M.G.	835.10
16	RASPUTIN, I.	815.50
39	KILOWATT, R.	766.05
65	VAN PELT, LUCY	701.16
48	CREEK, S. ANTONIO	623.20
58	SMITH	608.50
47	ZWICK, Z.	607.11
8	HEART, B.	450.25
28	BEACH, REDONDO	416.10
17	AARON, A.A.	410.05

IV. SYSTEM IMPLEMENTATION

This section of the paper contains information necessary for the user to program a dynamic storage job.

A. General Purpose Utility Subroutines

The routines described in this section are called by the system, however, they are also stand alone routines whose versatility may be capitalized upon by the user for debugging and or other purposes.

1. Subroutines BITX and UNPX

BITX (XKERN, II, JJ, WORDS)

BITX takes bits II through JJ of WORDS and stores them into XKERN filling the rest of XKERN (if any) with zeros. Bits are numbered with the right end of the word being bit 1 and the left end bit 60. JJ must not be greater than II+59 (i.e. not more than one word wide). Bits may be taken from at most two consecutive words. For example:

CALL BITX(XUSE, 122, 181, W)

would cause bits 2 through 60 of word W(3) and bit 1 of W(4) to be placed into word XUSE.

UNPX is an entry in subroutine BITX and functions exactly as BITX does.

20

2. Subroutine PACX

   PACX (XKERN, II, JJ, WORDS)

   PACX takes JJ-II+1 bits from the right end of XKERN and packs them into bits II through JJ of WORDS, leaving the rest of WORDS undisturbed. JJ must not be greater than II+59 (i.e. not more than 1 word wide). PACX undoes what BITX does.

3. Subroutine RDRUM

   RDRUM(LUN, TOWORD, NWORDS, IADRES)

   RDRUM reads NWORDS (1 word records) from the disc beginning with disc address IADRES into core location beginning with TOWORD. LUN is not used.

4. Subroutines RITEF, RITEI, RITEA, AND RITEO

   RITEF (IALPHA, DATA, NUMBER)

   RITEF prints one word IALPHA of BCD identification and DATA(I) words of information where I runs from 1 to NUMBER using a G20.7 format.

   RITEI functions as RITEF except that DATA is printed with an I10 format.

   RITEA functions as RITEF except that DATA is printed with an A10 format.

   RITEO functions as RITEF that DATA is printed with an Ø25 format.

5. Subroutine WDRUM

   WDRUM(LUN, FRMWRD, NWORDS, IADRES)

   WDRUM writes NWORDS (1 word records) onto the disc with disc address beginning at IADRES starting from core location FRMWRD. LUN is not used.

6. Subroutine XMIT

   XMIT (N,A,B)

   XMIT is a core to core transmission subroutine. If N is positive, N words are transferred from array A to array B. If N is negative, the variable A(1) is placed in cells B(1), B(2), ... B(N).

21

B. System Blank Common

The following variables are in the system blank common area:

QNAREA*	The number of dataset storage areas.
QWAREA(10)*	Word size of each dataset storage area.
QFREHD**	Header variable for the free block list.
QNDTST	The count of the current number of datasets in in existence.
QNLNKS	The count of the current number of list links in existence.
QZSIZE	The number of words in a system block.
QZHEAD	The header variable for the sytem block.
QCOUNT(30)	Counts the number of times system subroutines have been called:

```
QCOUNT(1)--CREATE
      (2)--CREATX
      (3)--DSTROY
      (4)--QGCBLK
      (5)--QGDBLK
      (6)--QGRBAG
      (7)--PUTBOT
      (8)--PUTTOP
      (9)--NEXT
     (10)--REMOVE
     (11)--NEXTNL
     (12)--PREV
     (13)--PREVNL
     (14)--PUTAFT
     (15)--PUTBEF
     (16)--LOCKDS
     (17)--NLOKDS
     (18)--LOCKFL
     (19)--NLOKFL
     (20)--CREATL
     (21)--QDSRED (COUNT OF DISC READS)
     (22)--QDSRYT (COUNT OF DISC WRITES)
     (23)--REMOVP
     (24)--PUTORD
     (25)--PUTOID
     (26)--PUTORA
     (27)--PUTOIA
     (28)--NOT USED
     (29)--NOT USED
     (30)--NOT USED
```

QDSIZE(10)*	Minimum dataset size.
QNSIZE**	Number of non-zero elements in QDSIZE array.
QLUNIT(10)*	Logical unit numbers of dataset storage areas.
QERLUN	Not used.
QFBITS(2,10)**	Bit numbers defining extent of data fields used in packed data words.
Q(1)	First location of dataset storage.
IQ(1)	Equivalenced to Q. Integer reference.
QQ(1,1)	Equivalenced to Q.
IQQ(1,1)	Equivalenced to Q.

All variables in blank common except Q and QQ are integer variables.

*Variables which must be set by the user.
**Variables which are set by subroutine QINITL.

C. System Subroutines

All system subroutines not accessible to the programmer and all system variables in blank common begin with the letter Q and are 6 characters in length.

Subroutines QDSRED and QDSRYT which read and write the disc can easily be modified to use other storage devices (such as a drum) should they become available.

APPENDIX I
LIST OF SYSTEM SUBROUTINES

The following system subroutines are meant to be accessed by the user. The following conventions are used to name variables:

```
LIST    --    List header variable
LINK    --    Link word
INDEX   --    Dataset index
LOCKNR  --    Lock number
INDEX 2 --    Dataset index
NWORDS  --    Number of words
IDSPWD  --    DSP word address
NWORD   --    Word number
KDSTRY  --    Dataset destroyed flag
```

1. Initialization

   CALL QINITL

2. Dataset assignment and release

   CALL CREATE (NWORDS, INDEX)
   CALL CREATL (NWORDS, INDEX)
   CALL CREATX (NWORDS, INDEX)
   CALL DSTROY (INDEX)

3. List manipulations

   CALL PUTTOP (LIST, INDEX)
   CALL PUTBOT (LIST, INDEX)
   CALL PUTORA (LIST, INDEX, NWORD)
   CALL PUTOIA (LIST, INDEX, NWORD)
   CALL PUTORD (LIST, INDEX, NWORD)
   CALL PUTOID (LIST, INDEX, NWORD)
   CALL PUTAFT (LIST, LINK, INDEX, INDEX 2)
   CALL PUTBEF (LIST, LINK, INDEX, INDEX 2)
   CALL NEXT (LINK, INDEX)
   CALL NEXTNL (LINK, INDEX)
   CALL PREV (LINK, INDEX)
   CALL PREVNL (LINK, INDEX)
   CALL REMOVE (LIST, LINK, INDEX)
   CALL REMOVP (LIST, LINK, INDEX)
   CALL WIPOUT (LIST, KDSTRY)

4. Dataset manipulations

   CALL DSLNTH (INDEX, NWORDS)
   CALL DSXPND (INDEX, NWORDS)
   CALL PUTDRM (INDEX)

5. Lock number manipulations

    CALL LOCKDS (INDEX, LOCKNR)
    CALL NLOKDS (INDEX)
    CALL LOCKFL (LIST, LOCKNR)
    CALL NLOKFL (LIST, LOCKNR)

6. DSP Word to/from Dataset Index

    CALL DSPWRD (INDEX, IDSPWD)
    CALL INDWRD (IDSPWD, INDEX)

The following system subroutines may be used by the user.

BITX	--	Unpacks a data word
UNPX	--	Unpacks a data word
PACX	--	Packs a data word
RDRUM	--	Reads the disc
RITEF	--	Prints labelled list
RITEA	--	Prints labelled list
RITEI	--	Prints labelled list
RITEO	--	Prints labelled list
WDRUM	--	Writes the disc
XMIT	--	Transmits core to core

The following system routines are not meant to be used by the user (except QINITL) but act as support programs to the system.

QCEASE	--	Writes error message and terminates
QCREAT	--	Locates a memory block of proper size for a dataset create operation
QDSRED	--	Performs all reading of datasets from disc
QDSRYT	--	Performs all writing of datasets to disc
QERROR	--	Writes error message and returns
QFIELD	--	A function which calls BITX to unpack data values from specified word fields.
QFLDST	--	Calls PACX to pack data values into specified word fields.
QGCBLK	--	Attempts to find a memory block of specified size by writing unlocked datasets to the disc
QGDBLK	--	Attempts to locate an available core on the disc of a specified size
QGRBAG	--	Performs garbage collection
QGTZWD	--	Obtains next available system word
QINITL	--	System initialization routine
QPTZWD	--	Returns system word to system availability list
QZBLOK	--	Creates and initializes system availability list

APPENDIX II
SYSTEM ERROR MESSAGES AND DEBUGGING AIDS

A. Error Messages

The following error messages are considered non-fatal. Execution continues but an unusual event has occurred.

ROUTINE	MESSAGE
DSTROY	ATTEMPT TO DESTROY A WELL-LOCKED DS. An attempt to destroy a dataset with lock number > 5.
PUTAFT	CANNOT LOCATE REQUESTED LIST ELEMENT. A call to PUTAFT or PUTBEF with LINK=0 and the program cannot locate a link of the list which corresponds to dataset INDEX.
QDSRED	READ ATTEMPT OUT OF LIMITS. An attempt to read from a dataset storage address greater than exists. Possible system error.
QDSRYT	WRITE ATTEMPT OUT OF LIMITS. Similar to above error message.
QGCBLK	UNLOCKED DATASET HAS NO DSP WORD. In the process of moving unlocked datasets to the disc a dataset with no DSP word has been encountered. This implies it belongs to no lists and therefore if moved cannot be referenced again.
QGRBAG	UNLOCKED DATASET HAS NO DSP WORD. See above.
REMOVE	CANNOT LOCATE REQUESTED LIST ELEMENT. A call to REMOVE (or REMOVP) with LINK=0 has occurred and the program is unable to locate a link of the list corresponding to dataset INDEX.

The following error messages are considered fatal. Execution is terminated after the message is printed.

ROUTINE	MESSAGE
QGCBLK	QGCBLK CALLED WITH QNAREA=1. Core storage memory has been filled and no disc storage has been provided for overflow.
QGCBLK	CANNOT GENERATE ENOUGH CORE SPACE. A block of size NWORDS cannot be fit into core storage after writing onto the disc. Probably too many locked datasets exist.
QGDBLK	NO DISC BLOCK AVAILABLE OF PROPER SIZE. No disc block large enough to hold the datasets to be moved exists.

B. Debugging Aids

The two routines discussed below provide debugging capability to the system user.

1. Subroutine DSADMP.

DSADMP (NWORDS)

DSADMP causes the first NWORDS (0 < NWORDS < 10) of each dataset to be printed (in octal) as well as the contents of all blank common cells. The dataset index, the lock number, and the total number of words in the dataset are also printed. If NWORDS=0 only the blank common cells are printed.

2. Subroutine LISPRT.

LISPRT (LIST, N)

LISPRT causes the index, number of words, lock numbers and the first four words to be printed for all datasets in the file specified by LIST (list header variable). The value of N specifies the list searching routine to be used:

N = 1 NEXT
    2 PREV
    3 NEXTNL
    4 PREVNL

ACKNOWLEDGEMENTS

The bulk of the coding* described in this paper was-performed by Mr. Ray L. Stone of General Research Corporation, Santa Barbara, California.

The technique of referencing dataset contents with FORTRAN names through the use of an equivalence block was developed at Compucenters, Incorporated, by Mr. Jim Wahl.

GLOSSARY OF TERMS

Dataset
: A group of contiguous computer storage cells containing related information or fields and treated as a unit. Also the data stored in the cells.

DSP Word
: A word created by the system for each dataset created to be part of a list. This word contains the current address of the dataset and the number of words in the dataset. DSP words are stored in system created words and updated by the system as datasets are moved within core storage and the disc.

Free Block
: A block of unused storage available for use.

Free List
: The list of free blocks.

Link Word
: A word created by the system for each list member containing the address of the next link word of the list, the previous link word of the list, and the address of the DSP word for the dataset. Link words are stored in system created blocks and automatically updated.

List
: A group of logically associated datasets connected by link words. Successive datasets in a list do not necessarily occupy adjacent memory blocks.

List Header Variable
: A word defined by the user containing the addresses of the first and last link words in the list. The variable name provides a means by which the user can refer to the list.

Lock Number
: A number ranging from 0 to 7 carried with each dataset to allow or inhibit movement within storage areas and release from the system.

Zeroth Word
: A word allocated by the system for each dataset in its first memory location. It contains the address of the DSP word of the dataset, the number of words in the dataset, and the lock number of the dataset. The user should not attempt to access this word.

DISTRIBUTION LIST

No. of Copies	Organization	No. of Copies	Organization
12	Commander Defense Documentation Center ATTN: DDC-TCA Cameron Station Alexandria, Virginia 22314	1	Commander U.S. Army Aviation Systems Command ATTN: AMSAV-E 12th & Spruce Streets St. Louis, Missouri 63166
1	Director Defense Advanced Research Projects Agency ATTN: Tech Info Ctr 1400 Wilson Boulevard Arlington, Virginia 22209	1	Director U.S. Army Air Mobility Research and Development Laboratory Ames Research Center Moffett Field, California 94035
1	Director Institute for Defense Analyses 400 Army-Navy Drive Arlington, Virginia 22202	1	Commander U.S. Army Electronics Command ATTN: AMSEL-RD Fort Monmouth, New Jersey 07703
1	Director Defense Intelligence Agency Washington, DC 20301	1	Commander U.S. Army Missile Command ATTN: AMSMI-R Redstone Arsenal, Alabama 35809
1	Commander U.S. Army Materiel Command ATTN: AMCDL 5001 Eisenhower Avenue Alexandria, Virginia 22333	1	Commander U.S. Army Tank Automotive Command ATTN: AMSTA-RHFL Warren, Michigan 48090
1	Commander U.S. Army Materiel Command ATTN: AMCRD, MG S. C. Meyer 5001 Eisenhower Avenue Alexandria, Virginia 22333	2	Commander U.S. Army Mobility Equipment Research & Development Center ATTN: Tech Docu Cen, Bldg. 315 AMSME-RZT Fort Belvoir, Virginia 22060
1	Commander U.S. Army Materiel Command ATTN: AMCRD, Dr.J.V.R.Kaufman 5001 Eisenhower Avenue Alexandria, Virginia 22333	1	Commander U.S. Army Armament Command Rock Island, Illinois 61202
1	Commander U.S. Army Materiel Command ATTN: AMCRD-T 5001 Eisenhower Avenue Alexandria, Virginia 22333		

31

DISTRIBUTION LIST

No. of
Copies   Organization

1   Commander
      U.S. Army Harry Diamond
        Laboratories
      ATTN: AMXDO-TI
      Washington, DC 20438

1   Director
      National Bureau of Standards
      Department of Commerce
      Washington, DC 20234

Aberdeen Proving Ground

      Ch, Tech Lib
      Dir,. USAMSAA
        ATTN: Dr. J. Sperrazza
               Mr. L. Bain
               Mr. W. Wenger
      Dir, USAHEL
        ATTN: Dr. J. Weiss
               Dr. R. Bauer
      Cmdr, USATECOM